September 1988

Report No. STAN-CS-88-1222

Also Numbered KSL-88-62

Load Balancing for Massively-Parallel Soft-Real-Time Systems

by

Max Hailperin

Department of Computer Science

Stanford University

Stanford, California 94305

Knowledge Systems Laboratory
Report No. KSL-88-62

August 1985

Load Balancing for Massively-Parallel Soft-Real-Time Systems

Max Hailperin

Department of Computer Science
Stanford University
Stanford, CA 94305

To appear in condensed form in:
Frontiers '88: The Second Symposium on the Frontiers of Massively Parallel Computation

Load Balancing for Massively-Parallel Soft-Real-Time Systems

Max Hailperin*
Knowledge Systems Laboratory
Computer Science Department
Stanford University
Stanford, CA 94305

August 30, 1988

Abstract

Global load balancing, if practical, would allow the effective use of massively-parallel ensemble architectures for large soft-real-time problems. The challenge is to replace quick global communications, which is impractical in a massively-parallel system, with statistical techniques. In this vein, we propose a novel approach to decentralized load balancing based on statistical time-series analysis. Each site estimates the system-wide average load using information about past loads of individual sites and attempts to equal that average. This estimation process is practical because the soft-real-time systems we are interested in naturally exhibit loads that are periodic, in a statistical sense akin to seasonality in econometrics. We show how this load-characterization technique can be the foundation for a load-balancing system in an architecture employing cut-through routing and an efficient multicast protocol.

*To appear in condensed form in *Frontiers '88: The Second Symposium on the Frontiers of Massively Parallel Computation.* This material is based upon work supported under a National Science Foundation Graduate Fellowship. Any opinions, findings, conclusions or recommendations expressed in this publication are those of the author and do not necessarily reflect the views of the National Science Foundation. This work was also supported by DARPA Contracts F30602-85-C-0012 and MDA903-83-C-0335, NASA Ames Contract NCC 2-220-S1, Boeing Contract W266875, and Digital Equipment Corporation.

1 Introduction

Our research group, the Stanford Knowledge Systems Laboratory Advanced Architectures Project, is exploring the construction of massively-parallel, object-oriented, knowledge-based, soft-real-time signal-interpretation systems. It seemed clear early on that some sort of adaptive load-distribution scheme would be necessary to allocate resources to such dynamic systems. Otherwise, in order to assure acceptable real-time performance, the system could only be lightly loaded, and the large-scale signal-interpretation problems the massive parallelism was intended to allow would not be possible. The remainder of this section explains why we desire a scheme which globally balances loads by migrating objects, and how we can exploit the somewhat periodic nature of our systems' loads to do global balancing in a manner appropriate to thousands of processing elements.

Much discussion in the load-distribution literature recently has centered on the choice of load balancing *vs.* load sharing [14]. While load balancing strives to keep all sites equally loaded, load sharing merely tries to prevent unnecessary idleness. Load balancing is appropriate to object-oriented real-time systems because

- real-time systems need to prevent long waits for processing—load balancing, by reducing the variance as well as the average of waiting times better achieves this; also,
- migrating objects to balance current load tends to also balance the future arrival of additional work at sites.

Traditionally, decentralized adaptive load-balancing systems have been local: they balance loads in small neighborhoods (the neighborhoods may be logical, rather than physical), and rely on repeated local adjustments to achieve global balance. (For a clear example, see the descriptions of diffusion in [12,13].) We find this inappropriate to our circumstances because

- modern interconnection networks employing cut-through or wormhole routing reduce the importance of locality [7],
- local techniques can fall prey to oscillation and wave-front-like propagation in the face of non-ideal conditions, and
- local techniques have difficulty responding quickly enough for dynamic and time-critical systems.

A global load-balancing system must somehow allow each site to estimate the current (or near-future) system-wide total load, in order that it may acquire or jettison sufficient work to bring its own load to the system-wide average. This seems incompatible with the constraints of a massively-parallel system: a site in a massively-parallel system must wait a considerable time to acquire global knowledge.

This apparent contradiction can be reconciled by using a stochastic time-series model to use prior load information to predict current loads. However, this approach is useless in most computer systems, as their loads are not very predictable.

Luckily, the real-time systems we are interested in (and many others) exhibit a different behavior. Their loads are periodic—not rigidly so, but rather in the same loose, statistical sense as many economic variables are seasonal. This periodicity is induced by sampled or scanned inputs and by sample-to-sample or scan-to-scan consistency in the outside world. Periodicity makes the loads more predictable, at least for lead times not greater than the period. As the period is generally relatively long, each site can have complete knowledge of loads at least through one period ago. This allows reasonably accurate prediction of current (or near-future) system-wide loads.

Notice that the statistical nature of this approach makes it appropriate to massively-parallel systems with thousands of processing elements:

- The large number of sites makes more straightforward methods employing global communications impractical.
- On the other hand, the large number of sites is necessary to make the statistical methods valid.

We are not suggesting this approach for real-time systems which are rigidly periodic; more direct use can be made of their periodicity. For example, Yan's "post-game analysis" method [17] could be used to successively refine a quasi-static mapping.

2 An Example Time Series

In this section we examine the evolution over time of the system-wide load in one of our real-time systems—an aircraft tracking and classification system [16]. We show that a simple stochastic model reasonably approximates this time series, that it is consistent with a common-sense understanding of the

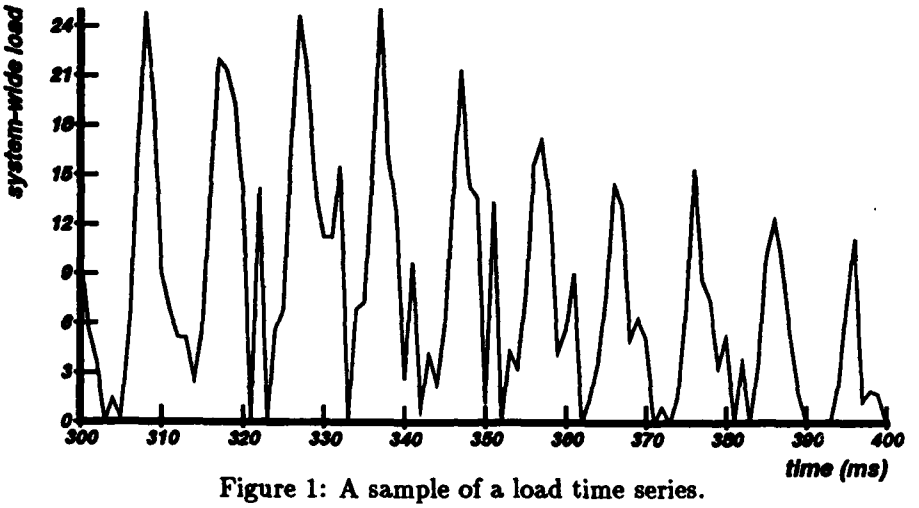

Figure 1: A sample of a load time series.

system, and that it allows moderately accurate prediction without recent complete information. Two notes are in order:

- Only the earliest, simplest, most data-driven stage of the system was operational when this data was taken; this results in a more regular time series than would otherwise be the case. In particular, diagnostic tests show our model to be incomplete, in that it misses a couple of sub-periods caused by the structure of the computation. We expect the structure of a complete system to be complex enough not to show through in the load time series.

- The plots in Figures 1 and 4 below show only a typical interval out of the larger time series which was analyzed.

Figure 1 shows the load over ten periods; each period is ten time quanta long, and the load value for each quantum is an average total of task queue lengths over that quantum. Notice that the pattern gradually shifts from period to period. Also, notice that as the observed activity diminishes, the system's performance varies from not quite keeping up with the input to having a relatively long period of quiescence between cycles. It is characteristic of real-time systems that they are sized so as to perform acceptably during peak periods, even if this means idleness at other times; this allows the periodicity of the input to show through as a periodicity of the load.

The sub-periods referred to above are also visible in the graph—the coarse sampling and small excerpt obscure it somewhat, but each major peak is followed by two smaller peaks whose sizes correlate with each other and that of the major peak.

2.1 Stochastic model

We analyzed this series using the methods of Box and Jenkins [3][1], and identified as a suitable first-cut model for it a multiplicative integrated moving average (IMA) process of orders $(0,1,1) \times (0,1,1)_{10}$. This model has the form:

$$z_t = z_{t-1} + z_{t-10} - z_{t-11} + a_t - \theta a_{t-1} - \Theta a_{t-10} + \theta\Theta a_{t-11},$$

where z_t is the system-wide load, a_t is a white-noise series, and θ and Θ are parameters. The structure of this process is more evident when written using the backwards shift operator B:

$$(1 - B)(1 - B^{10})z_t = (1 - \theta B)(1 - \Theta B^{10})a_t.$$

Adding the constraint that loads must be non-negative improves this basic model.

This model, while suggested by statistical evidence, is also plausible in terms of the mechanism of the system. The non-periodic component of the model essentially states that the load persists, except that it is subject to random perturbations. Some fraction (θ) of each random perturbation is of short-term effect only, while the remainder lasts until counteracted; this fits well with a birth-death view of processes. The periodic component of the model is identical in form, and can be similarly justified: the aircraft under observation (and thus the load pattern) remain constant except for random perturbations, some fraction $(1 - \Theta)$ of which are long-lasting entries or departures from the field of observation.

This model belongs to the broad class of stochastic processes known as ARMA (autoregressive–moving average) processes. It is interesting to ask why this particular ARMA process should be chosen—might others not fit as well? The answer is partially that this is the simplest periodic ARMA process whose periodic and non-periodic components are both:

- non-stationary (i.e., they have no fixed level),

[1] The equations in this section are reproduced with minor changes in notation from [3].

- stable (i.e., they don't grow explosively), and

- homogeneous (i.e., everywhere self-similar except for level).

Naturally a higher-order process could be used, which would fit better. However, it is generally preferable to use the simplest suitable model. Another possibility would be to drop the requirement of level independence by expanding the model to include a stationary autoregressive operator, i.e. by making it ARIMA (autoregressive–integrated moving average) rather than merely IMA. It can be argued that a busier system will spawn more processes, or alternatively that a busier system will run more processes to completion. We left this component out of our model because

- in a loaded system, the activity is not proportional to the load (as additional load means additional waiting tasks, rather than additional running tasks), and

- the statistical evidence does not unambiguously suggest such a component.

Diagnostic tests, as suggested by Box and Jenkins, showed that the model was only roughly fitting, due in part to the unmodeled sub-periods. This is especially evident in the cumulative periodogram of residuals, reproduced in Figure 2; the bulge around frequency 0.25 (period 4) shows that the model misses some periodicity in that neighborhood. (A cumulative periodogram shows an integrated power spectrum. A perfectly fitting model would leave white-noise residuals with a flat power spectrum and hence a straight diagonal cumulative periodogram.) Even the cumulative periodogram of ideal white-noise residuals might, because of the limited sample size, deviate outside the dashed lines approximately 25% of the time (the limit lines are calculated from the Kolmogorov-Smirnov test). Therefore, as the bulge just reaches the 25% limit line, it can't be considered an especially serious failure of the model. On the other hand, other statistical evidence and our understanding of the system indicate that the model is genuinely incomplete, rather than the bulge merely being an artifact of the limited sample size. We felt that incorporating these sub-periods into the model would be artificial, however, both because they are an artifact of the simplicity of the sample system, and also because they are not *a priori* known (or necessarily constant), unlike the externally imposed period.

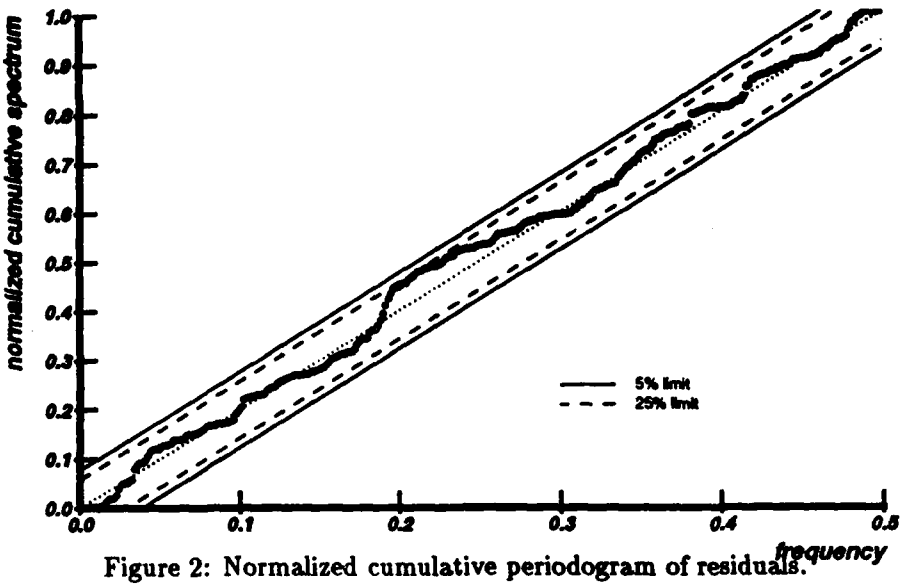

Figure 2: Normalized cumulative periodogram of residuals.

2.2 Forecasting

The non-periodic component of the model is that which is conventionally used for aperiodic computer systems; it gives rise to the familiar exponentially-weighted average forecast function. The periodic component in effect adds an exponentially-weighted average of corrections to this forecast, derived from the experience at corresponding points in earlier periods. Formally, the best one-step-ahead forecast possible for the model is found by assigning weights π_j to the loads j steps earlier, where

$$\begin{aligned}
\pi_j &= \theta^{j-1}(1-\theta),\ j = 1,\ldots,9 \\
\pi_{10} &= \theta^9(1-\theta) + (1-\Theta) \\
\pi_{11} &= \theta^{10}(1-\theta) + (1-\theta)(1-\Theta) \\
\pi_j &= \theta\pi_{j-1} + \Theta\pi_{j-10} - \theta\Theta\pi_{j-11},\ j \geq 12.
\end{aligned}$$

Depending on the relationship between θ and Θ, the heaviest weight in the forecast may either be on the most recent value, or on the one a period ago. In the aircraft tracking case (and many others, we speculate), there is more consistency from period to period than from instant to instant (as aircraft are more inertial than processes). This leads to the weights

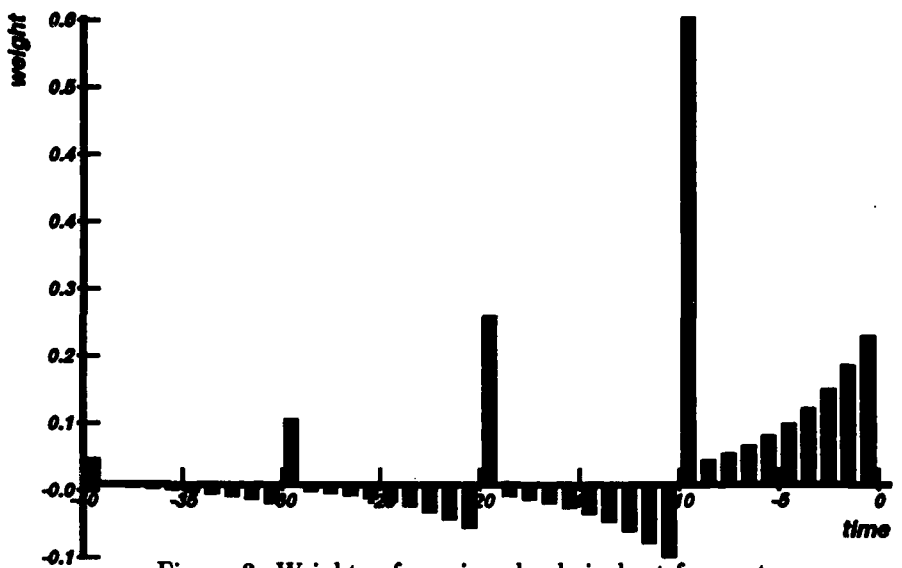

Figure 3: Weights of previous loads in best forecast.

illustrated in Figure 3, which were computed from the values for θ and Θ that best fit our sample series.

Forecasts can also be computed directly from the difference equation we used to define the model. In either case, forecasts for greater lead times can be calculated by repeated use of the step-ahead formula. (By lead time we mean the time from when the total load is last known to when the forecast is for.)

Since the period (in this case, the scan time of a radar) is long relative to the communication latencies of the system, it is reasonable to suppose that each site can have complete knowledge of all other sites' loads at least up until one period earlier, with diminishing knowledge thereafter. It should be possible in principle to make some use of the more recent, incomplete, information to improve the forecast, given a model of the load distribution with load balancing. In the next section we address this problem and show a heuristic solution. However, Figure 4 shows that even forecasts made using only data up through one period in advance are usually moderately accurate.

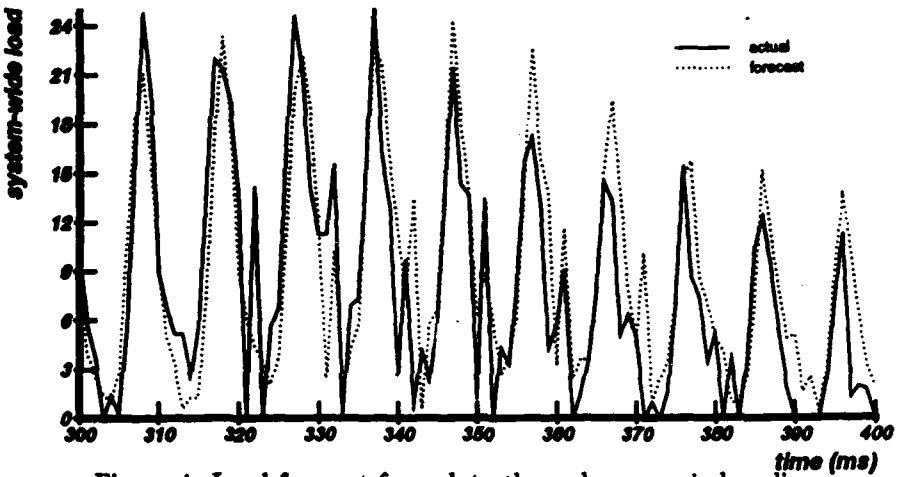

Figure 4: Load forecast from data through one period earlier.

2.3 How typical is this example?

Though this section presented a case study of a single time series taken from a single application, we believe the basic features are common to other systems as well. Preliminary results from experimentation with a passive radar interpretation system [4] confirm this belief. The IMA $(0,1,1) \times (0,1,1)_p$ model used here may well suit many such systems, though its suitability should of course be tested in each case. As well as testing the suitability of the model to a particular application, it is necessary to tune the parameters using sample time series. Systems with more than one period, for example from heterogeneous sensors, would necessitate a straightforward extension of the model.

One potential stumbling block in generalizing this technique to more realistic systems is that higher-level processing tends to be triggered by significant changes in the input (or by the lack of expected changes), rather than by the input itself. For example, a system that not merely tracks aircraft, but also attempts to deduce possible objectives, would reconsider the objective of an aircraft that sharply turned, or that failed to turn when it was expected to. This reduces the scan-to-scan consistency of the load. It remains to be seen how troublesome this is; clearly this depends on how much of the processing is special-case. When this issue came up in a discussion with a group familiar with actual systems, the consensus was that the load

on present-day systems is indeed quite periodic [15].

3 Incorporating Incomplete Information

The simple stochastic model presented in the preceding section only allows load information old enough to be complete (i.e. available from all sites) to be used. In this section we refine our model to allow incomplete information (i.e., more recent loads from some sites) to be employed. We formulate the problem, show an exact but impractical solution, and then present provably good practical heuristic approximations.

3.1 The problem

In order to understand what use a site can make of recent but incomplete information, we must refine our model to include how the system-wide total load is divided among the N sites. A simple, plausible version of this is to assume that the sites are independent instantaneously, but in the longer-term are successfully balanced. Formally, the model we have in mind is

$$z_{i,t} = a_{i,t} + \frac{z_{t-1} + z_{t-10} - z_{t-11} - \theta a_{t-1} - \Theta a_{t-10} + \theta\Theta a_{t-11}}{N},$$

where we use $z_{i,t}$ for the load of site i at time t (with $z_t = \sum_i z_{i,t}$) and similarly for $a_{i,t}$ and a_t (the $a_{i,t}$ are independently normally distributed, with variance σ_a^2).

As long as all $z_{i,t}$ are known, the $a_{i,t}$ can be calculated, and thus used for forecasting. When the information is incomplete, the deviation of the known $z_{i,t}$ from the step-ahead forecasts can no longer be attributed solely to their corresponding $a_{i,t}$, but rather will also include the persistent fraction of earlier unknown perturbations. The problem is to find the expected division between these two sources of perturbation, as the expected value of each $a_{i,t}$ should be incorporated into the forecast in its own way.

3.2 Exact solution

This problem can be solved by applying Bayes's theorem:

- We are given as a prior distribution for the $a_{i,t}$ that they are independently normally distributed with some variance σ_a^2.

- We make observations which imply a joint likelihood for the $a_{i,t}$ that is uniform where certain linear combinations of them (given below) equal the known $z_{i,t}$ and zero elsewhere.
- We would like to find the posterior joint distribution of the $a_{i,t}$, specifically its expected value, for use in forecasting.

The non-zero regions of the likelihood function can be found by rewriting the equation for $z_{i,t}$ in terms of the $a_{i,t}$ alone, using the summation operators $S = (1 + SB)$ and $S_{10} = (1 + S_{10}B^{10})$:

$$z_{i,t} = a_{i,t} + \frac{((1-\theta)SB + (1-\Theta)S_{10}B^{10} + (1-\theta)(1-\Theta)SS_{10}B^{11})a_t}{N}.$$

The posterior distribution can readily be written using Bayes's theorem, provided one is willing to leave some messy integrals in it. Unfortunately, this leaves numerical integration as the only way to find the needed expected value. This seems to be too much work to expect a load-balancing system to perform each time interval. What is needed is a pre-posterior analysis—a general analysis done in advance, into which specific numbers can be plugged at run time. Unfortunately, we know of no such approach to this problem in the general case. In the next subsection we consider heuristic approximations appropriate to our intended implementation. The analysis above serves as the standard by which the heuristics are judged, as well as suggesting them.

3.3 Heuristic approximations

The simplest heuristic is to simply assume that the full deviation of each known load $z_{i,t}$ from its step-ahead forecast is purely its corresponding $a_{i,t}$. This heuristic is actually the truth (given our model) for the first time-quantum with incomplete information, and can be shown to be a conservative approximation provided there is less than a period of incomplete information. By a conservative approximation, we mean that this heuristic is guaranteed to be more accurate than simply ignoring the incomplete information. This is because mistaking the retained portion of prior perturbations for current perturbation leads to it's being erroneously re-multiplied by $(1 - \theta)$, i.e. underestimated.

We can improve this approximation by taking advantage of one feature of our intended implementation. The implementation we suggest in section 5 uses a randomized style of information spreading known as "rumor mongering" which spreads each site's load information to an exponentially widening

fraction of the other sites. Thus the amount of load information a site has drops off exponentially with recency, and only the earliest incomplete load information is of any real significance.

In particular, for realistic parameters (e.g. a spreading factor of eight) the only significant improvement that could be made in the above simple heuristic would be to better account for the deviations observed in the second incomplete-information time-quantum. Moreover, this division between the first two incomplete-information time-quanta need not make use of information from later time-quanta, as such information would be very weak under these assumptions. This leaves a tractable two-quanta version of the general problem of the preceding subsection.

The $a_{i,t}$ from the N_n non-reporting sites of the first quantum can be lumped together, as can those from the N_r reporting sites of the second quantum. This is because of the symmetry amongst them. We will call the contribution of the former to the second-quanta deviations X and that of the latter Y. Our prior distributions for them are independent, normal, both have mean zero, and (by elementary probability theory) have the variances

$$\begin{aligned} \sigma_x^2 &= \frac{N_r^2}{N^2}(1-\theta)^2 N_n \sigma_a^2 \\ \sigma_y^2 &= N_r \sigma_a^2. \end{aligned}$$

We know that X and Y sum to the observed deviation, δ, of the second-quanta loads from their step-ahead forecasts. Therefore, the posterior distribution from Bayes's theorem gives us the following posterior expected values:

$$\begin{aligned} E(X) &= \frac{\int_{-\infty}^{\infty} x e^{-x^2/2\sigma_x^2 - (\delta-x)^2/2\sigma_y^2} dx}{\int_{-\infty}^{\infty} e^{-x^2/2\sigma_x^2 - (\delta-x)^2/2\sigma_y^2} dx} \\ &= \delta \frac{\sigma_x^2}{\sigma_x^2 + \sigma_y^2} \\ E(Y) &= \delta \frac{\sigma_y^2}{\sigma_x^2 + \sigma_y^2}. \end{aligned}$$

Thus we can readily at run time use the observed values of δ, N_n, and N_r to calculate a very good approximation to the best forecast possible with the available information.

4 Precision of Forecasts

In this section we analyze the potential for practical utility of our load-characterization scheme. We show that for the large numbers of sites characteristic of massively-parallel architectures, our scheme provides load estimates which are accurate enough to be useful for load balancing.

We can use the model of section 2 to calculate probability limits of forecasts—that is, the region around the forecast in which the actual system-wide load will lie some specified fraction of the time. Additionally, the more detailed model of section 3 specifies how the individual sites' loads can be expected to be distributed about the system-wide average load. What is most interesting is combining these two, in order to determine

- what fraction of the sites can be expected to be over- or under-loaded at some significance level, and
- how much relative error can be expected in the amount of work transferred between sites, due to erroneous forecasts.

Happily, we show that the accuracy of the forecasts relative to the standard-deviation of the site loads goes up with the square-root of the number of sites, so that for massively-parallel systems the uncertainty in the forecasts is unproblematic (assuming the validity of the model).

4.1 Probability limits of forecasts

The conditional probability distribution of the system-wide load about its forecast value is simply the sum of those of the a_t not included in the forecast. The error in the forecast will thus be normally distributed with mean zero and variance increasing with lead-time. For the IMA $(0,1,1) \times (0,1,1)_p$ model, if the forecast is made using complete information only, with lead time $l < p$, the variance is

$$V(l) = (1 + (l-1)(1-\theta)^2)N\sigma_a^2.$$

We can use the above formula to calculate approximate probability limits for the forecasts by substituting an estimate for σ_a. One approach would be to estimate it using the sample standard deviation from prior runs. Prior to the introduction of load balancing, the detailed model of section 3 certainly doesn't apply, but the system-wide model of section 2 presumably does, at least approximately. Therefore, the sample variance of the system-wide load

should be used as an initial estimate for $N\sigma_a^2$, rather than starting with the sample variance of individual site loads.[2] If the system-wide load sample standard deviation is s, then we can estimate that with probability ϵ the actual load differs from the lead l forecast by more than

$$u_{\epsilon/2} s \sqrt{1 + (l-1)(1-\theta)^2},$$

where $u_{\epsilon/2}$ is the $\epsilon/2$-tail-area point of the unit normal distribution. Notice that these bounds are for the total load—the standard deviation, and hence probability limits, for the average load are smaller by a factor of N.

4.2 Comparison with the distribution of site loads

Our model asserts that the loads of the individual sites at any time are normally distributed about the system-wide average load with standard deviation σ_a. We can compare this with the standard deviation of the lead l conditional probability distribution of the average load, which we derived in the previous subsection. The latter is larger by a factor of

$$\frac{\sqrt{1 + (l-1)(1-\theta)^2}}{\sqrt{N}};$$

the factor of $\sqrt{N}$ results from averaging N independent deviates.

This implies that for large systems the forecasts will be accurate enough to be useful. For example, if the system of section 2 could be spread among 1024 sites, even one-period-ahead forecasts would have a factor of 27 lower standard deviation than the site loads. Thus virtually all apparent over- or under-loads would be statistically significant, and the relative error in the amount of work transferred would be small (roughly 1/27).

5 Load-balancing Mechanism

In this section we outline a load-balancing scheme employing the load-characterization methodology of the preceding sections. Our scheme relies on a "rumor mongering" style of information spreading [9], which is appropriate to our architecture. We show that the mechanism not only allows sites to assess their load with respect to the system-wide average, but also

[2] We only wrote the formula in terms of the per-site σ_a^2 in order to be notationally consistent with section 3.

allows overloaded sites to reliably find sufficiently underloaded sites to which objects can be migrated.

If each site stores its knowledge of all sites' load histories, then they can spread their information around by a process of "rumor mongering"—that is, by randomly sharing information [10,1,2,9]. Naturally, the histories can be compressed by discarding information old enough to be scarcely relevant and by combining together loads from all sites where they all are known. Some information may be young enough to relevant to forecasting, but old enough to be well-known. This information can be retained but not passed on; [9] has a good discussion of such issues.

Our CARE ensemble architecture [8] uses a cut-through interconnection network, so latency is not proportional to distance (in the absence of contention). Additionally, it supports an efficient multicast protocol [5]. Therefore, we suggest that the information spreading be achieved by each site periodically multicasting its information to a random sample of the other sites. While the number of sites that each site will hear from in any given period varies, it can be shown that the distribution (a binomial distribution, rapidly approaching a Poisson distribution) is such that a paucity of information will be rare, even with a quite moderate sample size, e.g. eight.

Upon receiving a load-information message, a site should integrate the information into its own knowledge, and then use the time-series model (provided *a priori* based on experiments with the particular system) to estimate the current system-wide average load with probability limits. It should then compare this predicted average with its own current load, and with the load of the sender at the time of the sending. If the recipient appears significantly underloaded and the sender appears significantly overloaded, a request for work should be sent back.

This is a combination of random gossiping to distribute the information needed to decide whether and how much work to transfer, together with polling/bidding to match up the participating sites. As with all bidding schemes, some precautions are needed to avoid races. The underloaded site should not place any other requests for work until it receives work or an apology from the overloaded site. As the inter-arrival time for messages from overloaded sites should be high relative to the round-trip message time, few conflicts should occur.

The bidding could be reversed (overloaded sites could ask underloaded sites to accept work), but this would require that an extra message be sent. The system as we present it can best be classified as receiver-initiated [11], though in a sense the sender initiates the process by multicasting its load

information. This confusion of terminology results from our integration of the global-information-spreading and partner-seeking components of the mechanism.

It should be rare that an overloaded site cannot find enough total underload among the sites it samples to match its own overload. For example, suppose that the loads are normally distributed (as they are in the model of section 3), and that the sample size is eight. Of the eight sites sampled, it can be expected that four will be underloaded. The expected value of the absolute value of a normal deviate is $2/\sqrt{2\pi}$, or about .8 standard deviations, so the four underloaded sites will on the average have approximately 3.2 standard deviations worth of underload. But the originating site must really be far out on the tail of the distribution to have more than 3.2 standard deviations worth of overload. Notice that it is impossible to make as strong a statement in the reverse direction—this is an additional reason to favor a receiver-initiated transfer (it is more important for overloaded sites to reliably find underloaded sites than the converse).

The only aspect of load balancing not addressed by this mechanism is the choice of which objects to migrate. Here again the real-time nature of the system must be addressed. In general neither the highest- nor lowest-priority objects are best migrated, so as to neither unfairly advance a low-priority object nor hold up (due to migration time) a high-priority object. Chang addresses these issues in [6].

6 Acknowledgments

Anoop Gupta, Bruce Delagi, Harold Brown, and John Hennessy provided valuable feedback on this work. The entire Advanced Architectures Project made this work possible, by providing the technical context; many members additionally helped me with numerous specific difficulties. In this context, I'd like to specifically thank Greg Byrd, Bruce Delagi, Sayuri Nishimura, Alan Noble and Nakul Saraiya.

References

[1] Yeshayahu Artsy, Hung-Yang Chang, and Raphael Finkel. Processes migrate in Charlotte. Technical Report 655, Computer Sciences Department, University of Wisconsin–Madison, August 1986.

[2] Amnon Barak and Amnon Shiloh. A distributed load-balancing policy for a multicomputer. *Software—Practice and Experience*, 15(9):901–913, September 1985.

[3] George E. P. Box and Gwilym M. Jenkins. *Time Series Analysis: Forecasting and Control.* Holden-Day Inc., 1976.

[4] Harold D. Brown, Eric Schoen, and Bruce A. Delagi. An experiment in knowledge-based signal understanding using parallel architectures. Technical Report STAN-CS-86-1136, Department of Computer Science, Stanford University, October 1986.

[5] Gregory T. Byrd, Russell Nakano, and Bruce A. Delagi. A dynamic, cut-through communications protocol with multicast. Technical Report STAN-CS-87-1178, Department of Computer Science, Stanford University, September 1987.

[6] Hung-Yang Chang. Dynamic scheduling algorithms for distributed soft real-time systems. Technical Report 728, Computer Sciences Department, University of Wisconsin–Madison, 1987.

[7] William J. Dally. Wire-efficient VLSI multiprocessor communications networks. In *Advanced Research in VLSI: Proceedings of the 1987 Stanford Conference*, pages 391–415. The MIT Press, 1987.

[8] Bruce A. Delagi, Nakul Saraiya, Sayuri Nishimura, and Greg Byrd. An instrumented architectural simulation system. In *Artificial Intelligence and Simulation: The Diversity of Applications.* The Society for Computer Simulation International, February 1988.

[9] Alan Demers, Dan Greene, Carl Hauser, Wes Irish, John Larson, Scott Shenker, Howard Sturgis, Dan Swinehart, and Doug Terry. Epidemic algorithms for replicated database maintenance. In *Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing*, pages 1–12, August 1987.

[10] Zvi Drezner and Amnon Barak. A probabilistic algorithm for scattering information in a multicomputer system. Technical Report CRL-TR-15-84, Computing Research Laboratory, University of Michigan, March 1984.

[11] Derek L. Eager, Edward D. Lazowska, and John Zahorjan. A comparison of receiver-initiated and sender-initiated adaptive load sharing. *Performance Evaluation*, 6(1):53–68, March 1986.

[12] Robert H. Halstead, Jr. and Stephen A. Ward. The MuNet: A scalable decentralized architecture for parallel computation. In *Proc. 7th Annual Symposium on Computer Architecture*, pages 139–145, May 1980.

[13] Paul Hudak and Benjamin Goldberg. Experiments in diffused combinator reduction. In *1984 ACM Symposium on Lisp and Functional Programming*, pages 167–176, August 1984.

[14] Phillip Krueger and Miron Livny. Load balancing, load sharing and performance in distributed systems. Technical Report 700, Computer Sciences Department, University of Wisconsin–Madison, August 1987.

[15] Personal communication, September 10, 1987. Discussion with members of MIT Lincoln Laboratories Machine Intelligence Group.

[16] Russell Nakano and Masafumi Minami. Experiments with a knowledge-based system on a multiprocessor. Technical Report STAN-CS-87-1188, Department of Computer Science, Stanford University, October 1987.

[17] Jerry C. Yan. Managing and measuring two parallel programs on a multiprocessor. Technical Report CSL-TR-87-333, Computer Systems Laboratory, Stanford University, June 1987.